Knowledge Graph Embeddings for Dealing with Concept Drift in Machine Learning

Jiaoyan Chen[a], Freddy Lécué[b], Jeff Z. Pan[c,*], Shumin Deng[d,e], Huajun Chen[d,e]

[a]*Department of Computer Science, University of Oxford, UK*
[b]*INRIA, France; CortAIx@Thales, Canada*
[c]*School of Informatics, The University of Edinburgh, UK*
[d]*College of Computer Science and Technology, Zhejiang University, China*
[e]*ZJU-Alibaba Joint Lab on Knowledge Engine, China*

Abstract

Stream learning has been largely studied for extracting knowledge structures from continuous and rapid data records. However, efforts to understand whether knowledge representation and reasoning are useful for addressing concept drift[1], one of the core challenges from the stream learning community, particularly those due to dramatic changes in knowledge, have been limited and scattered. In this work, we propose to study the problem in the context of the semantic representation of data streams in the Semantic Web, i.e., ontology streams. Such streams are ordered sequences of data annotated with an ontological schema. A fundamental challenge is to understand what knowledge should be encoded and how it can be integrated with stream learning methods. To address this, we show that at least three levels of knowledge encoded in ontology streams are needed to deal with concept drifts: *(i)* existence of novel knowledge gained from stream dynamics, *(ii)* significance of knowledge change and evolution, and *(iii)* (in)consistency of knowledge evolution. We propose an approach to encoding such knowledge via schema-enabled knowledge graph embeddings through a combination of novel

*Corresponding author

Email addresses: `jiaoyan.chen@cs.ox.ac.uk` (Jiaoyan Chen), `freddy.lecue@inria.fr` (Freddy Lécué), `j.z.pan@ed.ac.uk` (Jeff Z. Pan), `231sm@zju.edu.cn` (Shumin Deng), `huajunsir@zju.edu.cn` (Huajun Chen)

[1]This work is addressing the challenge of concept drift in machine learning as opposed to concept drift in the Semantic Web community where “concept” (class) meaning in ontology TBox shifts from versioning, iterations or modifications. Note that changes in ABox alone can also lead to concept drift in learning from ontology streams.

representations: entailment vectors, entailment weights, and a consistency vector. We illustrate our approach on supervised classification tasks. Our main findings are that: *(i)* It is possible to develop a general purpose framework to address concept drifts in ontology streams by coupling any machine learning classification algorithms with our proposed schema-enabled knowledge graph embeddings method; *(ii)* Our proposed method is robust to significant concept drift (up to 51% of stream update ratio) and out-performs state of the art methods with 12% to 35% improvement on the Macro-F1 score in the tested scenarios; *(iii)* Only a small part of the ontological entailment (less than 20%) play an important role in determining the consistency between two snapshots; *(iv)* Predictions with consistent models outperform those with inconsistent models by over 300% in the two use cases. Our findings could help future work on applications of stream learning, such as autonomous driving, which demand high accuracy of stream learning in the presence of sudden and disruptive changes.

Keywords: Ontology, Stream Learning, Concept Drift, Knowledge Graph, Semantic Embedding

---

1. Introduction

On Context of Work: Stream learning, or the problem of extracting and predicting knowledge from evolving data, has been widely applied and largely studied. One typical application is forecasting the air quality index in the future with streaming observations of air pollutants, meteorological elements, traffic conditions and so on [1]. Other applications include stock price prediction, traffic monitoring, machine diagnostics, etc. Most techniques from the Database community, such as those using association rules [2], focus on syntactic representation of data to identify frequent associations and exploit them for prediction. Approaches in Machine Learning (ML) focus on learning prediction models such as random forests and Artificial Neural Networks for classifying or regressing data from streams in real-time [3, 4].

On Limitations so far: Although highly scalable and quite accurate, most ML approaches have been shown to be non robust to changes of statistical properties of the target variable, which the model is trying to predict. This is referred as the problem of *concept drift* in the ML community [5] as changes occur over time in unforeseen ways. Existing ML approaches such as those based on some temporal statistic measures [6] build models on old data, but knowledge inconsistencies can occur as time passes and the models may loss the effectiveness.

... and Tentatives so far: Towards this challenge different strategies such as online active learning, priority on recent data and dynamic sliding windows, have been proposed [4] (cf. more details in Related Work). Although such strategies can manage gradual changes, they fail in maintaining high accuracy (and other quantitative measures such as Precision, Recall or F1 Score) for sudden, disruptive changes. This is mainly due to the inconsistent evolution of the knowledge and the lack of metrics to understand the evolution of semantics. The current methods are mostly based on measures to statistic changes of the raw data.

On the Objective: This work aims at capturing unique properties of data streams to better detect, qualify and predict the concept drift, thus enhancing the model for stream learning.

On A New Context to Address the Initial Challenges: In this work, we propose to study the problem in the context of the semantic representation of data streams in the Semantic Web, i.e., ontology streams [7, 8]. Such streams are ordered sequences of data annotated with an ontological schema, where knowledge representation languages such as Web Ontology Language (OWL)[2] are used for modeling the semantics. From knowledge materialization [9] to predictive reasoning [10, 11], to textual explanations of reasoning results [12], to knowledge augmented transfer learning [13, 14], all are inferences where dynamics, semantics of data are exploited for deriving a priori knowledge from pre-established (certain) statements. However, inductive reasoning and ML have rarely been integrated to deal with ontology streams, not to mention dealing with the concept drift problem in learning from ontology streams. This significantly limits the discovery of additional knowledge from ontology streams.

On the Opportunity of such New Context: Representation learning [15], which refers to a variety of techniques that learn new representations of the raw data as more effective prediction input (features), has been widely investigated in different domains, such as natural language processing (e.g., BERT [16]) and computer vision (e.g., Graph Neural Networks [17]). From the Semantic Web perspective, components of knowledge base, such as entities, relations and facts can be represented by vectors, where their semantics such as the relationship to the neighbours is kept in the vector space. Despite that many of these techniques are known as knowledge graph embedding techniques [18], they are mainly for schema-less knowledge graphs until recently. The vector representations can be fed into down-

[2]https://www.w3.org/TR/2012/REC-owl2-overview-20121211/

stream statistical analysis and ML algorithms to discover new knowledge such as plausible facts and rules [19]. Although representation learning provides a way to understand the semantics of data for learning, there are currently few studies investigating the embeddings of expressive OWL ontologies, especially for the streaming context.

On Our Method: In order to incorporate semantics in stream learning to ensure accurate prediction, we propose an approach to encoding the structured knowledge into compact vector representations through learning from not only declared relationships between entities but also knowledge gained from materialization. The learned representations (embeddings) capture the relevant semantics to support reasoning tasks: *(i)* entailment to derive knowledge from axioms and rules, and *(ii)* consistency checking for capturing the validity of temporal evolution. Such schema-enabled embeddings are exploited in a context of supervised stream learning to learn prediction models, which are robust to concept drifts, including sudden and inconsistent changes. Note that our solution of incorporating reasoning in learning embeddings could also be used to solve other complex hybrid learning and reasoning tasks.

On Benefits and Our Contributions: This work first investigated the benefits of embedding semantics of data streams to tackle the problem of concept drift in stream learning. We presented a novel approach to embed the semantics of such streams, and particularly to embed core properties defining concept drifts: *(i)* the existence of novel knowledge gained from stream dynamics, *(ii)* the significance of knowledge change and evolution, and *(iii)* the consistency and inconsistency of knowledge evolution. Such knowledge is embedded through a combination of novel representations: entailment vectors, entailment weights, and a consistency vector. Such schema-enabled knowledge graph embeddings capture the concept drift properties, and are better fit-for-purpose when trying to tackle the problem of concept drift. Finally developed a consistent prediction framework that is adaptable and flexible to basic ML models such as Logistic Regression, for dealing with concept drift. We illustrate our approach on classification tasks for supervised steam learning. The experiments have shown its higher classification macro-averaged F1 score in comparison with the state-of-the-art approaches on bus delay forecasting and air quality index forecasting with real world data streams from Dublin in Ireland and Beijing in China.

On the Rest of the Paper: The next two sections review the related work, the adopted logic and the ontology stream learning problem. In Section 4 we introduce the concept drift. Section 5 presents our methodology including the knowledge graph embedding and the prediction. Section 6 presents the experiments and evaluation. Section 7 concludes the paper.

2. Related Work

The related work includes *(i)* stream learning with concept drift, *(ii)* ontology stream reasoning, and *(iii)* representation learning for the Semantic Web (i.e., knowledge graph embedding).

2.1. Stream Learning with Concept Drift

There have been several methods for addressing the concept drift problem in stream learning. We classify them into three categories.

• **Recent Priority:** These methods assign higher weights to more recent samples or more recently trained models. For example, Cao et al. [20] trained adaptive Support Vector Machine (SVM) models by placing higher weights on the errors of the more recent training samples. For another example, Chu et al. [21] utilized online active learning with customized properties of weighting. In these methods, long-term historical samples are sometimes discarded, which is known as the forgotten mechanism.

• **Dynamic Sliding Window:** In these methods, a dynamic-size window of recently seen samples are kept by monitoring the data change over time, based on which the model can be updated accordingly [6]. One typical algorithm is ADaptive Windowing (ADWIN), proposed in [22]. It acts as a change detector or estimator by evaluating the model's error, so as to adjusting the window size.

• **Model Ensemble:** These methods often train multiple models and combine their prediction results. One typical example is Adaptive-Size Hoeffding Tree Bagging proposed by Bifet et al. [23]. It trains multiple classification trees with different segments of historical data and assigns an adaptive weight to each tree by monitoring its error. A similar sample segmentation and model ensemble technique was also adopted by Gao et al. [24] for streaming data with a skewed distribution. Combing multiple models have been shown to perform better than one single (complex) model in many real world applications [3].

For high performance, the above strategies are often used together. Ensemble learning can be integrated with drift detection algorithms, and often incorporates dynamic updates, such as selective removal or addition of models, where the principle of recent priority and the data change detector are sometimes adopted [25]. Leverage Bagging [26] is a state-of-the-art ensemble-based stream learning algorithm that updates models by detecting data changes with ADWIN.

The recent priority strategy assumes that the temporally adjacent data is more representative information for prediction [24], but this assumption is violated in case of sudden and disruptive changes. Change detectors with a dynamic sliding window use some intermediate metrics e.g., model error [22] for measurement. This, however, leads to additional biases. Combing multiple models can outperform one single model, but it often relies on change detection or recent priority assumption for model selection, removal and addition [3]. Last but not the least, these statistical methods ignore domain knowledge (such as the semantics of variables) and overall context (e.g., relationship between variables across streams) in understanding changes and concept drifts.

2.2. Ontology Stream Reasoning

Stream reasoning, or materialization over dynamic knowledge has been widely studied and applied [27, 28]. Some of these studies focus on continuous query over semantic data in less expressive languages RDF/RDFS (e.g., C-SPARQL [29]), while the others provide capabilities of semantic query, entailment reasoning and rule-based reasoning for evolving OWL ontologies (e.g., TrOWL [30, 31] and sigRL [32]). A successful industry application is turbine diagnosis in Siemens, where ontology-based data access, semantic query and reasoning are conducted jointly for the analysis of streaming sensor data [32].

To the best of our knowledge, there are few studies investigating (predictive) analysis over ontology streams. [33] proposed a novel formalization of predicting future knowledge for streaming data represented by a Description Logic (DL) family DL-*Lite*, where rules are mined to represent complex data association patterns. [10] and [11] proposed to predict future or missing knowledge over ontology streams represented by DL $\mathcal{EL}^{++}$, where consistent snapshots are first inferred with semantic reasoning, and semantic association rules are further mined and selected for prediction. [34] improved the scalability of the above method with incremental, approximate maintenance algorithms.

In these predictive analytics studies for ontology streams (a.k.a. predictive reasoning) [33, 10, 11, 34], the problem of concept drift however is not considered. They depend on semantic rules to represent data patterns and infer new knowledge, which makes them incompatible to popular ML models such as logistic regression and random forests. Meanwhile, the scalability and efficiency issues, caused by rule mining and selection limited their applications.

2.3. Knowledge Graph Embedding

Representation learning aims at learning representations of the raw data so as to make it easier to extract useful information in downstream statistical or predictive analysis [15]. It has been recently applied in the Semantic Web. Components of a knowledge base such as entities and relations, are embedded into a vector space with their semantics such as their relative relationships kept, for downstream mining and prediction tasks such as entity categorization, link prediction, question answering and rule learning [18, 19, 35]. These techniques are known as knowledge graph embedding, as they are originally developed for knowledge graphs which are knowledge bases composed of facts in RDF form. A variety of knowledge graph embedding methods have been proposed, including those tensor factorization based (e.g.,RESCAL [36]), embedding translation based (e.g., TransE [37]) and neural language model based (e.g., RDF2Vec [35]).

Most of these embedding methods, as far as we know, are still limited to knowledge bases composed of RDF facts alone, while those OWL ontologies or knowledge bases with ontological schemas have not been widely investigated. Some methods such as JOIE [38] support ontology-aware embeddings, but the ontology is often simple, expressed by RDF schema instead of any more expressive languages such as OWL. Paulheim and Stuckenschmidt [39] predicted the consistency of an expressive ontology by representing its ABox axioms with a feature vector that is composed of binary values. Different from [39], our embedding method learns the weights of ABox axioms.

Although there have been some studies for embedding OWL ontologies (e.g., EL Embedding [40] and OWL2Vec* [41]), they are limited to a static context which is totally different from the streaming context we aim at in this study. The knowledge graph embedding method in this paper on the one hand utilizes the expressiveness of OWL ontologies in two ways: inferring underlying entailments for richer semantics and checking consistency between two axiom sets, on the other hand explores the context of streaming ontologies. It infers entailments for OWL ontology streams, encodes them into vectors by learning the weights of entailments and checking the consistency between snapshots, and further applies all these embeddings for robust steam learning and prediction.

3. Background

In this section, we present the settings of our work, and how data stream, ontology and ontology stream are all connected to set-up the scene of a stream learning problem where concept drift is the challenge to tackle.

3.1. Settings

Data streams are captured as temporal evolution of data through update. As we aimed at exploring the role of semantics for addressing the challenge of concept drift in stream learning, data (in streams) will be represented through semantic representation. In particular the semantics of data is represented using an ontology. We focus on Description Logic (DL) based ontologies since it offers reasoning support for most of its expressive families and is compatible to W3C OWL (2) standard. Therefore data streams with attached semantic representations are characterized as ontology streams. The core objective of the work is to understand whether uplifting data streams with semantic representation is a way forward to tackle the problem of concept drift in stream learning.

The next subsections expose details on *(i)* the semantics used for representation, *(ii)* the definition of ontology stream, and *(iii)* the characterization of the ontology stream learning problem. The definitions and numerous concepts are illustrated with examples from our experimental context in Dublin on bus delay forecasting. In particular the very first examples are important as they illustrate the semantic representation of data (Example 1), ontology streams (Example 2), streams change (Example 3 - which is the basics behind concept drift), ontology stream learning problem (Example 4). All remaining examples are illustrations of our novel definitions and concepts, which are required to tackle the problem of concept drift in ontology stream learning.

3.2. DL $\mathcal{EL}^{++}$

Our work is illustrated using DL $\mathcal{EL}^{++}$ [42], which is a sub-language of the tractable OWL 2 EL profile in the OWL 2 family. A signature Σ, denoted as $(\mathcal{N}_C, \mathcal{N}_R, \mathcal{N}_I)$, consists of disjoint sets of *(i)* atomic concepts $\mathcal{N}_C$, *(ii)* atomic roles $\mathcal{N}_R$, and *(iii)* individuals $\mathcal{N}_I$. Given a signature, the top concept $\top$, the bottom concept $\bot$, an atomic concept A, an individual a, an atomic role r, $\mathcal{EL}^{++}$ concept expressions C and D can be composed with the following constructors:

$$\top \mid \bot \mid A \mid C \sqcap D \mid \exists r.C \mid \{a\}$$

A DL ontology $\mathcal{O} \doteq \langle \mathcal{T}, \mathcal{A} \rangle$ is composed of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. Briefly a TBox is a set of concepts, roles and their relationship axioms. $\mathcal{EL}^{++}$ supports General Concept Inclusion axioms (e.g. $C \sqsubseteq D$), Role Inclusion axioms (e.g., $r \sqsubseteq s$). An ABox is a set of concept assertion axioms (e.g., $C(a)$), role assertion axioms (e.g., $R(a, b)$), and individual in/equality axioms (e.g., $a \neq b$, $a = b$).

Example 1. ***(TBox and ABox Concept Assertion Axioms)***
Figure 1 presents (i) a fragement of a TBox $\mathcal{T}$, where the concept $DisruptedRoad$ (2) *defines "roads which are adjacent to an event that causes high disruption", (ii) some concept assertions such as* (11) *and* (12) *which mean the road r_0 is adjunct to roads r_1 and r_2 respectively.*

$SocialEvent \sqcap \exists type.Poetry \sqsubseteq Event \sqcap \exists disruption.Low$					(1)
$Road \sqcap \exists adj.(\exists occur.\exists disruption.High) \sqsubseteq DisruptedRoad$					(2)
$Road \sqcap \exists adj.(\exists occur.\exists disruption.Low) \sqsubseteq ClearedRoad$					(3)
$BusRoad \sqcap \exists travel.Long \sqsubseteq DisruptedRoad$					(4)
$BusRoad \sqcap \exists travel.OK \sqsubseteq ClearedRoad$					(5)
$Road \sqcap \exists with.Bus \sqsubseteq BusRoad$	(6)			$Road(r_0)$	(7)
$Road(r_1)$	(8)	$Road(r_2)$	(9)	$Bus(b_0)$	(10)
$adj(r_0, r_1)$	(11)	$adj(r_0, r_2)$	(12)	$Long \sqcap OK \sqsubseteq \bot$	(13)

Figure 1: TBox Fragment and ABox Fragment of The Example Ontology.

All completion rules of DL $\mathcal{EL}^{++}$, which are used to classify individuals and entail subsumption, are described in [42]. Reasoning with such rules is PTime-Complete.

3.3. Ontology Stream

We represent knowledge evolution by dynamic and evolutive ontology versions [7]. In such a context all data (ABox) and entailments (inferred statements) are changing over time, while the schema (TBox) remains unchanged.

Definition 1. ***(DL $\mathcal{EL}^{++}$ Ontology Stream)***
A DL $\mathcal{EL}^{++}$ ontology stream $\mathcal{P}_m^n$ from point of time m to point of time n is a sequence of Abox axioms $(\mathcal{P}_m^n(m), \mathcal{P}_m^n(m{+}1), \cdots, \mathcal{P}_m^n(n))$ with respect to a static TBox $\mathcal{T}$ of DL $\mathcal{EL}^{++}$, where $m, n \in \mathbb{N}$ and $m < n$.

$\mathcal{P}_m^n(i)$ is also known as a snapshot of an ontology stream $\mathcal{P}_m^n$ at time i, referring to all ABox axioms at time i. A transition from $\mathcal{P}_m^n(i)$ to $\mathcal{P}_m^n(i{+}1)$ is seen as an ABox update. We denote by $\mathcal{P}_m^n[i, j]$, i.e., $\bigcup_{k=i}^{j} \mathcal{P}_m^n(k)$ a windowed stream of $\mathcal{P}_m^n$ between time i and j with $i \leq j$. Any window $[i, j]$ has a fixed length. Windows of length 1 are denoted as $[i]$. We consider streams $[\alpha] \doteq [i, j]$ and $[\beta] \doteq [k, l]$ ($0 \leq i < j \leq n$, $0 \leq k < l \leq n$) of $\mathcal{P}_0^n$ as two windows in $[0, n]$.

Example 2. ***(DL $\mathcal{EL}^{++}$ Ontology Stream)***
Figure 2 illustrates $\mathcal{EL}^{++}$ ontology streams $\mathcal{P}_0^n$, $\mathcal{Q}_0^n$, $\mathcal{R}_0^n$, related to events, travel time, buses, through snapshots at time $i \in \{0, 1, 2, 3\}$, i.e., a window on $[0, 3]$. Note n is any integer greater than or equal to 3 in our example. Their dynamic knowledge is captured by evolutive ABox axioms such as (20) *which captures e_1 as "a social event on poetry occurring in road r_2" at time 1.*

By applying completion rules on the static TBox $\mathcal{T}$ and the ABox sequence $\mathcal{P}_0^n$, snapshot-specific axioms are inferred. Namely, for each snapshot at time i, entailments are inferred with its ABox axioms $\mathcal{P}_0^n(i)$ and $\mathcal{T}$. In Definition 2 (from [11]), the evolution of a stream is captured along its changes, i.e., through *new*, *obsolete* and *invariant* ABox entailments from one windowed stream to another.

Definition 2. ***(ABox Entailment-based Stream Changes)***
Let $\mathcal{S}_0^n$ be an ontology stream; $[\alpha]$, $[\beta]$ be its windows in $[0, n]$; $\mathcal{T}$ be the static TBox, $\mathcal{G}$ be its snapshot-specific ABox entailments. The changes occurring from $\mathcal{S}_0^n[\alpha]$ to $\mathcal{S}_0^n[\beta]$, denoted by $\mathcal{S}_0^n[\beta]\nabla\mathcal{S}_0^n[\alpha]$, are ABox entailments in $\mathcal{G}$ being new (14)*, $obsolete$* (15)*, $invariant$* (16)*.*

$$\mathcal{G}_{new}^{[\alpha],[\beta]} \doteq \{g \in \mathcal{G} \mid \mathcal{T} \cup \mathcal{S}_0^n[\beta] \models g \ \wedge \mathcal{T} \cup \mathcal{S}_0^n[\alpha] \not\models g\} \tag{14}$$

$$\mathcal{G}_{obs}^{[\alpha],[\beta]} \doteq \{g \in \mathcal{G} \mid \mathcal{T} \cup \mathcal{S}_0^n[\beta] \not\models g \ \wedge \mathcal{T} \cup \mathcal{S}_0^n[\alpha] \models g\} \tag{15}$$

$$\mathcal{G}_{inv}^{[\alpha],[\beta]} \doteq \{g \in \mathcal{G} \mid \mathcal{T} \cup \mathcal{S}_0^n[\beta] \models g \ \wedge \mathcal{T} \cup \mathcal{S}_0^n[\alpha] \models g\} \tag{16}$$

(14) reflects knowledge we gain by sliding the window from $[\alpha]$ to $[\beta]$, while (15) and (16) denote respectively lost and stable knowledge. All duplicates are assumingly removed. Definition 2 provides basics, via ABox entailments [43], for understanding how knowledge evolves over time.

Example 3. ***(ABox Entailment-based Stream Changes)***
Table 1 illustrates changes occurring from $(\mathcal{Q}\cup\mathcal{R})_0^n[0, 1]$ to $(\mathcal{Q}\cup\mathcal{R})_0^n[2, 3]$ through ABox entailements. For instance, "r_2 as a disrupted road window $[2, 3]$ of $(\mathcal{Q} \cup \mathcal{R})_0^n$ is new with respect to knowledge in $[0, 1]$. It is entailed by axioms (4)*,* (6)*,* (9)*,* (24)*,* (25)*,* (27) *and* (28)*.*

3.4. Ontology Stream Learning Problem

Definition 3 revisits classic supervised learning problem [44] for ontology stream as the problem of predicting class assertion entailments in a future snapshot according to the current and historical snapshots.

$$\mathcal{P}_0^n(0) : (Incident \sqcap \exists impact.Limited)(e_3), \quad occur(r_1, e_3) \tag{17}$$
$$\mathcal{Q}_0^n(0) : (Road \sqcap \exists travel.OK)(r_1) \tag{18}$$
$$\mathcal{R}_0^n(0) : with(r_1, b_0) \tag{19}$$

$$\mathcal{P}_0^n(1) : (SocialEvent \sqcap \exists type.Poetry)(e_1), \quad occur(r_2, e_1) \tag{20}$$
$$\mathcal{Q}_0^n(1) : (Road \sqcap \exists travel.OK)(r_2) \tag{21}$$
$$\mathcal{R}_0^n(1) : with(r_2, b_0) \tag{22}$$

$$\mathcal{P}_0^n(2) : (Event \sqcap \exists disruption.High)(e_2), \quad occur(r_2, e_2) \tag{23}$$
$$\mathcal{Q}_0^n(2) : (Road \sqcap \exists travel.Long)(r_2) \tag{24}$$
$$\mathcal{R}_0^n(2) : with(r_2, b_0) \tag{25}$$

$$\mathcal{P}_0^n(3) : (Event \sqcap \exists disruption.High)(e_2), \quad occur(r_2, e_2) \tag{26}$$
$$\mathcal{Q}_0^n(3) : (Road \sqcap \exists travel.Long)(r_2) \tag{27}$$
$$\mathcal{R}_0^n(3) : with(r_2, b_0) \tag{28}$$

Figure 2: Ontology Streams $\mathcal{P}_0^n(i), \mathcal{Q}_0^n(i), \mathcal{R}_0^n(i)_{i\in\{0,1,2,3\}}$.

Windowed Stream	$(\mathcal{Q} \cup \mathcal{R})_0^n[2,3] \nabla (\mathcal{Q} \cup \mathcal{R})_0^n[0,1]$		
Changes	$obsolete$	$invariant$	new
$with(r_2, b_0)$		✓	
$ClearedRoad(r_2)$	✓		
$DisruptedRoad(r_2)$			✓

Table 1: ABox Entailment-based Stream Changes.

Definition 3. ***(Ontology Stream Learning Problem)***
Let $\mathcal{S}_0^n$ be an ontology stream; $\mathcal{T}$, $\mathcal{A}$ be its TBox and ABox respectively; $g \in \mathcal{G}$ be an ABox entailment; k be an integer in $(0, n]$. An Ontology Stream Learning Problem, noted OSLP$\langle \mathcal{S}_0^n, k, \mathcal{T}, \mathcal{A}, g\rangle$, is the problem of estimating whether g can be entailed from $\mathcal{T}$ and $\mathcal{A}$ at time k of stream $\mathcal{S}_0^n$, given knowledge at time $t < k$ of $\mathcal{S}_0^n$.

This estimation is denoted as $p_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n(k) \models g)$ with values in $[0,1]$ and $k \geq 1$. One estimation, adopted from [45], is directly using the ratio of previous snapshots where entailment g is entailed. In detail, it can be calculated as

$$p_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n(k) \models g) \doteq \frac{p_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^{k-1} \models g)}{p_{|\mathcal{T}\cup\mathcal{A}}(a \in \mathcal{S}_0^{k-1})} \tag{29}$$

where the estimation $p_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^{k-1} \models g)$ is the proportion of snapshots in $\mathcal{S}_0^{k-1}$ entailing g, while the estimation $p_{|\mathcal{T}\cup\mathcal{A}}(a \in \mathcal{S}_0^{k-1})$ is the proportion of snapshots that contain some class assertions of the individual a which is involved in g. Specially, $p_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n(k) \models g) \doteq 0$ if there are no snapshots that contain any class assertions of the individual a, i.e., $p_{|\mathcal{T}\cup\mathcal{A}}(a \in \mathcal{S}_0^{k-1}) = 0$. The conditional probability of a in $\mathcal{S}_0^{k-1}$ (i.e., $a \in \mathcal{S}_0^{k-1}$), given that $\mathcal{S}_0^{k-1}$ entails g, is 1.

Example 4. ***(Ontology Stream Learning Problem)***
The problem of estimating whether class assertion g, defined as $DisruptedRoad(r_2)$, can be entailed by $\mathcal{T}$ and $\mathcal{A}$ at time point 4 of $(\mathcal{Q} \cup \mathcal{R})_0^n$ is defined as OSLP$\langle(\mathcal{Q} \cup \mathcal{R})_0^n, 4, \mathcal{T}, \mathcal{A}, g\rangle$. The estimation can be retrieved using (29) *hence $p_{|\mathcal{T}\cup\mathcal{A}}((\mathcal{Q} \cup \mathcal{R})_0^n(4) \models DisruptedRoad(r_2)) \doteq 2/3$.*

In the above description of OSLP, the entailment g to be predicted is described as a class assertion (e.g., $DisruptedRoad(r_2)$), but it can also be a role assertion. This depends on how the domain data and prediction task are modeled with the ontology. Assume the ontology TBox has defined $Disrupted$ (a class of road condition), c_1 (an individual of $Disrupted$), and $hadRoadCond$ (a relation between a road and a condition), predicting the above entailment $DisruptedRoad(r_2)$ is equivalent to predicting the role assertion $hasRoadCond(r_2, c_1)$. To model a ML multi-class classification task by OSLP with class assertion, multiple class assertions in a snapshot should be predicted, and each of them corresponds to one class. In prediction, one score is calculated for each class assertion, and the class of the one with the largest score is adopted as the output class.

The (naive) example of estimation by (29) can also be replaced by other (more complex) models such as a linear score function. In our method, the simple estimation (29) is used for concept drift analysis, while its results together with the knowledge graph embeddings are further integrated in order to learn the linear score function for final prediction.

4. Concept Drift in Ontology Stream

In this section we introduce semantic concept drift, its significance and measures to quantify sudden and disruptive changes in an ontology stream.

4.1. Semantic Concept Drift

Definition 7 revisits concept drift [24] for ontology streams as *prediction changes* in ABox entailments (Definition 4), which include two types: *sudden changes* and *disruptive changes* (Definition 5 and 6).

Definition 4. ***(Prediction Change)***
Let $\mathcal{S}_0^n$ be an ontology stream; $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{G}$ be its TBox, Abox and ABox entailments. A prediction change in $\mathcal{S}_0^n$ is occurring between time i and j in $[0, n]$ with respect to $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{G}$ iff:

$$\exists g \in \mathcal{G} : \|p_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n(i) \models g) - p_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n(j) \models g)\| \geq \varepsilon \tag{30}$$

where $\varepsilon \in (0, 1]$ is a variable bounding the difference of estimation, $\|\cdot\|$ refers to the absolute value, and $i < j$.

The ABox entailment g is called an evidence entailment of the prediction change. $\mathcal{G}$ is the set of candidate evidence entailments. We adopt the classification, relation and property entailments of those named individuals that have already exist in the original ABox before entailment reasoning. Namely entailments involving individuals that are generated during the inference are excluded. We denote by $\mathbb{C}_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n, i, j, \varepsilon)$, the set of all evidence entailments of the prediction change with an ε difference between time i and j of ontology stream $\mathcal{S}_0^n$. Note as only the snapshots before time t are given for the OSLP problem, all the concept drift analysis in Section 4 by default only adopts the snapshots before time t (e.g., $i, j < k$ in Definition 4).

Example 5. ***(Prediction Change)***
$g \doteq DisruptedRoad(r_2)$ can be entailed from $\mathcal{T}$ and $\mathcal{A}$ at time 2 of $(\mathcal{Q} \cup \mathcal{R})_0^n$ with a zero probability following (29)*. Therefore a prediction change between times 2 and 4 (cf. Example 4) is captured with $g \in \mathbb{C}_{|\mathcal{T}\cup\mathcal{A}}((\mathcal{Q} \cup \mathcal{R})_0^n, 2, 4, 1/3)$.*

Definition 5. ***(α-Sudden Prediction Change)***
A prediction change at point of time i in stream $\mathcal{S}_0^n$, satisfying (30)*, is defined as α-sudden, with $\alpha \in (0, n-i]$ iff $j = i + \alpha$.*

Definition 6. ***(Disruptive Prediction Change)***
A prediction change, satisfying (30)*, is disruptive iff $\exists g' \in \mathcal{G}$ s.t.*

$$\mathcal{T} \cup \mathcal{A} \cup g \cup g' \bigcup_{l=0}^{\max\{i,j\}} \mathcal{S}_0^n(l) \models \bot \tag{31}$$

where $\bigcup_{l=0}^{\max\{i,j\}} \mathcal{S}_0^n(l)$ captures all axioms from any snapshot $\mathcal{S}_0^n(l)$ of stream $\mathcal{S}_0^n$ with $l \in [0, \max\{i, j\}]$, $\mathcal{G}$ is the same as in Definition 4.

Suddenness (Definition 5) characterises the proximity of prediction changes in streams. A lower α means closer changes. Disruptiveness (Definition 6) captures disruptive changes from a semantic perspective, i.e., conflicting knowledge among snapshots $\mathcal{S}_0^n(i)$, $\mathcal{S}_0^n(j)$ with respect to the knowledge $\mathcal{T} \cup \mathcal{A}$.

Definition 7. ***(Semantic Concept Drift)***
A semantic concept drift in $\mathcal{S}_0^n$, is defined as a 1-sudden prediction change or a disruptive prediction change.

Evaluating if a concept drift occurs for a snapshot update is in worst case polynomial time with respect to acyclic TBoxes and $\mathcal{S}_0^n$ in $\mathcal{EL}^{++}$ since subsumption and satisfiability in (30), (31) can be checked in polynomial time [42].

Example 6. ***(Semantic Concept Drift)***
Two prediction changes from time $i = 2$ to 3 and 3 to 4 (cf. Table 2) have occurred for $g \doteq DisruptedRoad(r_2)$ in $(\mathcal{Q} \cup \mathcal{R})_0^n$. They are semantic concept drifts as they are 1-sudden and disruptive with $g' \doteq ClearedRoad(r_2)$ in $(\mathcal{Q} \cup \mathcal{R})_0^n(1)$.

Prediction			Prediction Change	
Past Points of Time	Time i	$p_{\mid\mathcal{T}\cup\mathcal{A}}$ $((\mathcal{Q} \cup \mathcal{R})_0^n(i) \models g)$	$g \in \mathbb{C}_{\mid\mathcal{T}\cup\mathcal{A}}$ $((\mathcal{Q} \cup \mathcal{R})_0^n, i, i{+}1, {}^1\!/\!_3)$	Disruptiveness
$\{0\}$	1	0	✗	✗
$\{0,1\}$	2	0	✓	✓
$\{0,1,2\}$	3	$^1\!/\!_2$	✓	✓
$\{0,1,2,3\}$	4	$^2\!/\!_3$	N/A	N/A

Table 2: Prediction Changes in $(\mathcal{Q} \cup \mathcal{R})_0^n$ $(g \doteq Disrupted(r_2))$.

4.2. Significance of Concept Drift

The significance of a semantic concept drift (Definition 8) is an indicator on its severity. It captures the homogeneity of the concept drifts across ABox entailments as the proportion of ABox entailments from $\mathcal{S}_0^n(i)$ and $\mathcal{S}_0^n(i{+}1)$ causing semantic concept drifts. The value of the significance ranges in $[0, 1]$.

Definition 8. ***(Semantic Concept Drift Significance)***
The significance of a semantic concept drift, defined between points of time $i \in (0, n)$ and $i{+}1$ of $\mathcal{S}_0^n$ with ε, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{G}$ as the difference, TBox, ABox, and entailments, respectively, is $\sigma_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n, i, \varepsilon)$:

$$\frac{|\mathbb{C}_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n, i, i{+}1, \varepsilon)|}{|\{g \in \mathcal{G} \mid \mathcal{T} \cup \mathcal{S}_0^n(i) \models g \vee \mathcal{T} \cup \mathcal{S}_0^n(i{+}1) \models g\ \}|} \tag{32}$$

where $|\cdot|$ represents the cardinality of a set.

As Definition 7, calculating the semantic concept drift significance by (32) is in worst case polynomial time.

Example 7. ***(Semantic Concept Drift Significance)***
By applying (32) *on concept drifts of Table 2 we derive that* $\sigma_{|\mathcal{T}\cup\mathcal{A}}((\mathcal{Q}\cup\mathcal{R})_0^n, 2, {}^1\!/\!_3)$ *is* ${}^4\!/\!_7$ *while* $\sigma_{|\mathcal{T}\cup\mathcal{A}}((\mathcal{Q}\cup\mathcal{R})_0^n, 3, {}^1\!/\!_3)$ *is* 0*, hence there is a more significant drift between times* 2 *and* 3 *than between* 3 *and* 4*. In other words conflicting facts* $g \doteq DisruptedRoad(r_2)$ *and* $g' \doteq ClearedRoad(r_2)$ *w.r.t.* $\mathcal{T}$ *and* $\mathcal{A}$ *have the most significant impact on prediction changes at times* 2 *and* 3*.*

Remark 1. ***(Semantic Concept Drift Evolution)***
A semantic concept drift in any ontology stream $\mathcal{S}_0^n$ *is more significant at time* i *(*$i > 0$*) than at time* $i{+}1$ *if* $|\mathcal{G}_{new}^{[0,i],[0,i+1]}| = 0$*.*

Proof. (Sketch) Since $|\mathcal{G}_{new}^{[0,i],[0,i+1]}| = 0$, $\mathcal{S}_0^n(i)$ and $\mathcal{S}_0^n(i{+}1)$ are similar w.r.t. $\models_{\mathcal{T}\cup\mathcal{A}}$. Thus, the set of all entailments, predicted at $i{+}1$ and $i{+}2$ from (29), are similar but with different prediction values (30) $\forall\varepsilon \geq 0$. So $\sigma_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n, i, \varepsilon)$ and $\sigma_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n, i{+}1, \varepsilon)$ in (32) have same denominators while $\mathbb{C}_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n, i{+}1, i{+}2, \varepsilon) \subseteq \mathbb{C}_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n, i, i{+}1, \varepsilon)$ hence $\sigma_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n, i{+}1, \varepsilon) \leq \sigma_{|\mathcal{T}\cup\mathcal{A}}(\mathcal{S}_0^n, i, \varepsilon)$. $\square$

Algorithm 1 retrieves significant concept drifts in $\mathcal{S}_0^n$ with minimal significance $\sigma_{\min}$, where $\mathcal{G}$ refers to ABox entailments about classification, property and relation of named individuals that already exist in the original ABox. It iterates on all snapshot updates except those with no new ABox entailment (Line 5 - Remark 1) for minimizing satisfiability and subsumption checking. Semantic concept drifts, as 1-sudden and disruptive prediction changes, are retrieved (Line 7). Algorithm 1 completes the process (Line 9) by filtering concept drifts by the minimal significance $\sigma_{\min}$.

Computing the significant concept drifts with Algorithm 1, given an acyclic TBox and $\mathcal{S}_0^n$ in DL $\mathcal{EL}^{++}$, is in worst case polynomial time, due to the complexity of evaluating a semantic drift (cf. the complexity of Definition 7). The number of pairwise combinations of snapshots is quadratic w.r.t. the number of snapshots in the window that is considered. Therefore computing significant α-sudden, disruptive prediction changes following Algorithm 1 is also in worst case polynomial time.

Algorithm 1: `[A1]SignificantDrift`$\langle \mathcal{O}, \mathcal{S}_0^n, \varepsilon, \sigma_{\min} \rangle$

Input: (i) Axioms $\mathcal{O} : \langle \mathcal{T}, \mathcal{A}, \mathcal{G} \rangle$, (ii) Ontology stream $\mathcal{S}_0^n$, (iii) Lower limit $\varepsilon \in (0, 1]$ of prediction difference, (iv) Minimum threshold of drift significance $\sigma_{\min}$.
Result: $\mathbb{S}$: $\mathbb{S}$ignificant concept drifts in $\mathcal{S}_0^n$ w.r.t. $\sigma_{\min}$.
begin
  $\mathbb{S} \leftarrow \emptyset$; % *Init. of the* $\mathbb{S}$*ignificant concept drift set.*
  foreach $i \in (0, n]$ *of* $\mathcal{S}_0^n$ *such that* $|\mathcal{G}_{new}^{[0,i],[0,i+1]}| \neq 0$ **do**
    % *Selection of* 1*-sudden, disruptive prediction changes.*
    if $\exists g, g' \in \mathcal{G}$ *such that:*
      $\|p_{|\mathcal{T} \cup \mathcal{A}}(\mathcal{S}_0^n(i) \models g) - p_{|\mathcal{T} \cup \mathcal{A}}(\mathcal{S}_0^n(i{+}1) \models g)\| \geq \varepsilon$
      $\wedge\ \mathcal{T} \cup \mathcal{A} \cup \mathcal{S}_0^n(i) \cup \mathcal{S}_0^n(i{+}1) \cup g \cup g' \models \perp$ **then**
      % *Semantic concept drifts with min. significance.*
      if $\sigma_{|\mathcal{T} \cup \mathcal{A}}(\mathcal{S}_0^n, i, \varepsilon) \geq \sigma_{\min}$ **then**
        $\mathbb{S} \leftarrow \mathbb{S} \cup \{(i, i{+}1)\}$ % *Add snapshot update.*
  return $\mathbb{S}$;

5. Ontology Stream Learning

This section describes the approach towards ontology stream learning where the difficulty is incorporating the semantics of stream dynamics introduced in the above section into a supervised ML classification algorithm. To this end, we developed our knowledge graph embedding method (i.e., Algorithm 2). As a core element, this embedding algorithm represents entailments and the semantic consistency by vectors, based on which the sampling strategy and the sample weight update strategy of the downstream ML classification algorithm are developed. We finally present such embeddings can be used for prediction using any ML classification algorithm, and our overall prediction approach (Algorithm 3) is agnostic to them. Indeed the knowledge graph embeddings are the main representations that are required to encode and incorporate the semantics of stream dynamics, which are then utilized by a ML model for prediction.

5.1. Knowledge Graph Embeddings

The semantics of streams exposes three levels of knowledge which are crucial for learning with concept drifts: *(i)* existence of class assertion entailments inferred from stream assertions and axioms, *(ii)* significance of different entailments in comparing two snapshots, and *(iii)* consistency and inconsistency of knowledge evolution. Such semantics are encoded as knowledge graph embeddings including *entailment vectors*, *entailment weights*, and a *consistency vector*. The embeddings are based on not only original ABox assertions but also the underlying entailments inferred with TBox axioms, so that more complete semantics is captured.

Bag of Entailment (BOE) Encoding: Assume we have a set of candidate ABox entailments, denoted by E, consisting of concept and role assertions of those named individuals extracted from the snapshots of $\mathcal{S}_0^n$ before time k, with the size of d. The BOE encoding captures the presence and non presence of each ABox entailment in a given snapshot. Namely, a snapshot is represented by a binary value vector $\boldsymbol{b} = (b_1, b_2, ..., b_d)$, where b_i is set to 1 if entailment i is inferred, and is set to 0 otherwise. The vector is defined as BOE *entailment vector*, while this procedure is called BOE encoding, denoted by $\mathcal{B}$.

Let n_i, n_c and n_r be the number of unique individuals, concepts and roles respectively in the set E of candidate ABox entailment, the maximum value of the BOE dimension d is $n_i \times n_i \times n_r + n_i \times n_c$. In real word applications (cf. air quality forecasting and bus delay forecasting in Section 6), the dimension is, however, often much smaller than the maximum value due to the sparsity of the relations between individuals. It can also be configured according to the application by methods like filtering out those entailments that appear in only a small ratio of snapshots.

Weighted Bag of Entailment (WBOE) Encoding: We assume different entailments have different importance in comparing two snapshots. We define *entailment weights* as a vector $\boldsymbol{w} = (w_1, w_2, ..., w_d)$, where each weight $w_{i,1 \leq i \leq d}$ is a numeric value to measure the importance of an entailment in E. With entailment weights, the BOE encoding can be extended to weighted bag of entailment (WBOE) encoding: $\boldsymbol{e} = \boldsymbol{b} \cdot \boldsymbol{w}$, where $\cdot$ represents calculating dot product between two vectors, and $\boldsymbol{e}$ is defined as WBOE entailment vector. The similarity between two snapshots in vector space can be measured with Euclidean distance:

$$\phi(\boldsymbol{e_1}, \boldsymbol{e_2}) = \|\boldsymbol{e_1} - \boldsymbol{e_2}\|_2 = \|\boldsymbol{b_1} \cdot \boldsymbol{w} - \boldsymbol{b_2} \cdot \boldsymbol{w}\|_2 \tag{33}$$

where $\|\cdot\|_2$ represents calculating L2-norm.

Entailment Weight Learning: The entailment weights $\boldsymbol{w}$ aim to project the inter-snapshot similarity w.r.t. the prediction task into a vector space constrained by the entailments E. They are learned according to the change of the entailment g of the snapshot. To this end, we first define task-specific (in-)consistent snapshot pair (Definition 9), where the ground truth entailment $\bar{g}$ refers to the target entailment

g in current and past snapshots, whose truth is already known (either observed or inferred). $\boldsymbol{w}$ will also be learned based on these task-specific consistent and inconsistent snapshots.

Definition 9. ***(Task-specific (In-)Consistent Snapshot Pair)*** *Let $\mathcal{S}_0^n$ be an ontology stream, $\bar{g}$ be the ground truth entailment, Δ be the prediction time ($\bar{g}$ and Δ together is known as a task). Given two snapshots $\mathcal{S}_0^n(h)$ and $\mathcal{S}_0^n(t)$ with $h \leq t$, $\mathcal{S}_0^n(h)$ and $\mathcal{S}_0^n(t)$ are defined as a task-specific consistent snapshot pair if the change of g from $\mathcal{S}_0^n(h)$ to $\mathcal{S}_0^n(h+\Delta)$ is the same as that from $\mathcal{S}_0^n(t)$ to $\mathcal{S}_0^n(t+\Delta)$, and a task-specific inconsistent snapshot pair otherwise. $\mathcal{S}_0^n(h)$ is called a head snapshot while $\mathcal{S}_0^n(t)$ is called a tail snapshot.*

Example 8. ***(Task-specific (In-)Consistent Snapshot Pair)*** *Assume Δ is 1, $\bar{g}$ in snapshot $\mathcal{S}_0^n(0)$, $\mathcal{S}_0^n(1)$, $\mathcal{S}_0^n(2)$ and $\mathcal{S}_0^n(3)$ is $DisruptedRoad(r_2)$, $ClearedRoad(r_2)$, $DisruptedRoad(r_2)$ and $ClearedRoad(r_2)$ respectively, $\mathcal{S}_0^n(0)$ and $\mathcal{S}_0^n(2)$ are task-specific consistent snapshot pair, while $\mathcal{S}_0^n(1)$ and $\mathcal{S}_0^n(2)$ are task-specific inconsistent snapshot pair.*

We apply Algorithm 2 where the weights $\boldsymbol{w}$ are iteratively learnt through a training procedure which aims at minimizing the loss (34). Note that Algorithm 2 only reports the sampling strategy and the iterative procedure to minimize the loss and to obtain the weights that satisfy the loss. The training strategy could be achieved through a neural network architecture using classical feedforward and back-propagation mechanisms. We initially extract those task-specific consistent and inconsistent snapshot pairs from the current and past snapshots. One consistent pair corresponds to one positive sample $(\boldsymbol{e_h^+}, \boldsymbol{e_t^+})$, while one inconsistent pair corresponds to one negative sample $(\boldsymbol{e_h^-}, \boldsymbol{e_t^-})$, where $\boldsymbol{e_h}$ ($\boldsymbol{e_e}$ resp.) represents the WBOE vector of the head (tail resp.) snapshot. The training aims to minimize the following max-margin-based loss function:

$$O(\boldsymbol{w}) = \sum_{(\boldsymbol{e_h^+}, \boldsymbol{e_t^+}) \in \boldsymbol{S}^+} \sum_{(\boldsymbol{e_h^-}, \boldsymbol{e_t^-}) \in \boldsymbol{S}^-} max\left\{0, \gamma + \phi(\boldsymbol{e_h^+}, \boldsymbol{e_t^+}) - \phi(\boldsymbol{e_h^-}, \boldsymbol{e_t^-})\right\} \tag{34}$$

where γ is a fixhyperparameter for the margin, $\boldsymbol{S}^+$ and $\boldsymbol{S}^-$ represent the positive and negative sample sets respectively. Optimization algorithms like Stochastic Gradient Descent (SGD) [46] and Adam [47] can be adopted as the optimizer for training. The details for learning $\boldsymbol{w}$ are shown in Algorithm 2.

The setting of hyperparameters (b, μ, m, γ) and the optimizer could be adjusted. Indeed, Algorithm 2 is capturing the training process for computing knowledge graph embeddings which does require fine-tuning the hyperparameters. The one that makes the loss curve converge and finally lead to the minimum loss is adopted. In real applications, approximation with hyperparameter searching algorithms like grid search on best configurations for (b, μ, m, γ) as well as some empirical tricks [46] (e.g., stopping increasing the maximum iteration number m when the loss curve can already converge) are adopted.

As Algorithm 2 is exposing an iterative method to compute graph embeddings through the minimization of loss function (34), stochastic gradient descent could be used to derive the gradient and update the embeddings.

Example 9. ***(Entailment Weight)***
The example sequence $(\mathcal{Q} \cup \mathcal{R})_0^n$ include 5 ABox entailments: $ClearedRoad(r_1)$, $ClearedRoad(r_2)$, $DisruptedRoad(r_2)$, $with(r_1, b_0)$ and $with(r_2, b_0)$, with weights of 0.89, 0.92, 1.13, 0.02 and 0.12 respectively. The named classification entailments: $ClearedRoad$ have much higher importance than the relationship entailment: $with$ when comparing two snapshots w.r.t. bus delay.

Definition 10. ***(Consistency Vector)***
A consistency vector of snapshot $\mathcal{S}_0^n(i)$ in $\mathcal{S}_0^n$, denoted by $\boldsymbol{c}_i$, is defined $\forall j \in [0, n]$ by c_{ij} if $i < j$; c_{ji} otherwise such that:

$$c_{ij} \doteq \begin{cases} \frac{h(\mathcal{G}_{inv}^{i,j})}{h(\mathcal{G}_{new}^{i,j})+h(\mathcal{G}_{inv}^{i,j})+h(\mathcal{G}_{obs}^{i,j})} & \textit{if } \mathcal{T} \cup \mathcal{S}_0^n(i) \cup \mathcal{S}_0^n(j) \not\models \bot \\ \frac{h(\mathcal{G}_{inv}^{i,j})}{h(\mathcal{G}_{new}^{i,j})+h(\mathcal{G}_{inv}^{i,j})+h(\mathcal{G}_{obs}^{i,j})} - 1 & \textit{otherwise} \end{cases} \tag{35}$$

where the function $h(\cdot)$ calculates the sum of weights of new (14)*, obsolete* (15) *and invariant* (16) *ABox entailments from $\mathcal{S}_0^n(i)$ to $\mathcal{S}_0^n(j)$. Assume $\boldsymbol{b}$ is the BOE vector of an entailment set $\mathcal{G}$, then $h(\mathcal{G}) = |\boldsymbol{b} \cdot \boldsymbol{w}|$, where $|\cdot|$ calculates a vector's L1-norm. $c_{ij} = c_{ji}$ for $\forall i, j \in [0, n]$.*

A consistent vector with values in $[-1, 1]^{n+1}$, encodes *(i)* consistency (inconsistency resp.) with positive (negative resp.) values, and *(ii)* similarity of knowledge among $\mathcal{S}_0^n(i)$ and any other snapshot $\mathcal{S}_0^n(j)_{j\in[0,n]}$ of stream $\mathcal{S}_0^n$ w.r.t. axioms $\mathcal{T}$ and $\mathcal{A}$. In (35), a larger number of invariant entailments leads to higher similarity, while a larger number of new and obsolete ABox entailments, capturing some differentiators in knowledge evolution, leads to lower similarity. Meanwhile, entailments that have larger weights have higher impact. When a semantic inconsistency occurs, the value 1 is subtracted instead of considering its additive inverse. This ensures that the invariant factor has always a positive impact. The consistency vector also indicates the stability of an ontology stream.

Algorithm 2: `[A2]KnowledgeGraphEmbedding` $\langle b, \mu, m, \gamma, \mathcal{S}_0^n, E \rangle$

Input: *(i)* Mini-batch size: b, *(ii)* Learning rate: μ, *(iii)* Maximum iteration number: m, *(iv)* Margin: γ, *(v)* A sequence of snapshots, i.e., ontology stream: $\mathcal{S}_0^n$, *(vi)* A set of entailments: E with the size of d.
Result: $\boldsymbol{w} = (w_1, w_2, ..., w_d)$: weights for entailments E.
begin
  Uniformly initialize $\boldsymbol{w}$;
  Normalize $\boldsymbol{w}$: $w_i = \frac{w_i}{\|\boldsymbol{w}\|_2}$;
  Set iteration: $t = 0$;
  while $t \leq m$ **do**
    $\%$ *sample a batch of positive snapshot pair*
    $\boldsymbol{S}_{batch} \leftarrow positive\_sampling(\mathcal{S}_0^n, b)$;
    $\boldsymbol{T}_{batch} \leftarrow \emptyset$; $\%$ *initialize the input matrix*
    foreach $(\boldsymbol{h}^+, \boldsymbol{t}^+) \in \boldsymbol{S}_{batch}$ **do**
      $\%$ *sample a negative snapshot pair for each positive snapshot pair*
      $(\boldsymbol{h}^-, \boldsymbol{t}^-) \leftarrow negative\_sampling(\mathcal{S}_0^n, \boldsymbol{h}^+, \boldsymbol{t}^+)$;
      $\%$ *encode each snapshot to a vector by WBOE*
      $\boldsymbol{e}_h^+, \boldsymbol{e}_t^+, \boldsymbol{e}_h^-, \boldsymbol{e}_t^- \leftarrow encoding(\boldsymbol{h}^+, \boldsymbol{t}^+, \boldsymbol{h}^-, \boldsymbol{t}^-)$;
      $\boldsymbol{T}_{batch} = \boldsymbol{T}_{batch} \cup \big\{(\boldsymbol{e}_h^+, \boldsymbol{e}_t^+), (\boldsymbol{e}_h^-, \boldsymbol{e}_t^-)\big\}$;
    $\%$ *Update weights according to the derivative of* (34)
    Update $\boldsymbol{w}$: $\boldsymbol{w}$ += $\mu \sum_{((\boldsymbol{e}_h^+, \boldsymbol{e}_t^+), (\boldsymbol{e}_h^-, \boldsymbol{e}_t^-)) \in \boldsymbol{T}_{batch}} \nabla O(\boldsymbol{w})$;
    Normalize $\boldsymbol{w}$: $w_i = \frac{w_i}{\|\boldsymbol{w}\|_2}$;
    Set iteration: t += 1;
  return $\boldsymbol{w}$;

Example 10. ***(Consistency Vector)***
Consistency vector $\boldsymbol{c}_3$*, i.e.,* $(c_{03}, c_{13}, c_{23}, c_{33})$ *of* $(\mathcal{Q} \cup \mathcal{R})_0^n(3)$ *is* $(0, -0.94, 1, 1)$. *Knowledge at time* 3 *is consistent / inconsistent / similar with knowledge at times* 0 */* 1 */* 2 *and* 3.

The BOE embedding through entailment vector calculation and the evaluation of the consistency vector by (35) are in worst case polynomial time with respect to $\mathcal{T}$ and $\mathcal{S}_0^n$ in $\mathcal{EL}^{++}$ (ABox entailment in polynomial time [42]). Computation complexity of Algorithm 2, i.e., entailment weight calculation for WBOE embedding is $O(dbm)$, depending on the model complexity, i.e., the size of candidate ABox entailments d), batch size b and iteration number m.

5.2. Semantic Prediction

Algorithm 3 trains the final prediction model by minimizing the loss (36), with the above concept drift analysis and the learned embeddings. This learning process is applied on the $\mathbf{N}$ concept drifted snapshots from $\mathcal{S}_0^n$ before time k ($\mathbf{N} < k$), in order to infer a prediction at time k. These snapshots with concept drifts are denoted as $\mathcal{S}_0^n|_\kappa$, where κ refers to the proportion of these snapshots. Note $\mathcal{S}_0^n|_\kappa$ is selected to capture *(i)* $\mathcal{S}_0^n(k-1)$, i.e., the closest (temporally) to $\mathcal{S}_0^n(k)$ (Line 4), *(ii)* knowledge in the most (Line 8-9) significant concept drifts (Definition 8 - Line 5), *(iii)* any other snapshots to meet $\mathbf{N}$ – the expected size (Line 11). In other words the prediction model captures the temporal dependencies of snapshots through knowledge graph embeddings and aims at learning a compact representation of such dependencies through the embeddings. The computation of the model is driven by (36) as the task is to minimize it. The results of applying the model computed in Algorithm 3 are values (multivariate in case of multi-dimensional space) at the snasphot of time k. The model is strongly constrained by significant concept drift and consistency to account for such properties of the stream.

The model is trained with samples of the form $\{(\boldsymbol{x}_i, g_i) \mid i \in \{1, \ldots, \mathbf{N}\}\}$ where $\boldsymbol{x}_i$ is the concatenation of the WBOE vector $\boldsymbol{e}$ and the feature vector, i.e., data properties of the corresponding snapshot of i from $\mathcal{S}_0^n$ (i.e., $\mathcal{S}_0^n|_\kappa(i)$), and $\boldsymbol{v}(g_i)$ is the target variable in $\{0, 1\}$, where 1 indicates g_i is true (is observed or can be inferred) and 0 indicates the opposite. The model is represented by a linear scoring function $f(\boldsymbol{x}_i) = a^T\boldsymbol{x}_i + b$ with model parameters $a \in \mathbb{R}^{\mathbf{N}}$ and $b \in \mathbb{R}$. The goal of learning is to minimize the following objective function:

$$O_j(a, b) \doteq \sum_{i=1}^{\kappa} \omega_{ij} L(\boldsymbol{v}(g_i), f(\boldsymbol{x}_i)) + \alpha R(a), \tag{36}$$

where L represents a loss function (e.g., hinge and $\log$), R is a regularization term and $\alpha > 0$ is a non-negative hyperparameter. R and α together control the variance of the model in case of overfitting. Each sample $(\boldsymbol{x}_i, g_i)$ in (36) is weighted by ω_{ij} which is calculated either by (37) or (38). (37) (resp. (38)) which filters out consistent (resp. inconsistent) historical snapshots can be adopted for steams with a high (resp. low) percentage of snapshots with concept drifts. Note the concept drift with semantic concept drift significance can be detected via Definition 8). See Model Consistency Impact in Section 6 for more discussions and evaluation on the selection of (37) and (38).

$$\omega_{ij} \doteq \begin{cases} 0, & \text{if } c_{ij} > 0 \\ -c_{ij} & \text{else,} \end{cases} \quad (37) \qquad \omega_{ij} \doteq \begin{cases} 0, & \text{if } c_{ij} < 0 \\ c_{ij} & \text{else,} \end{cases} \quad (38)$$

Algorithm 3: `[A3]PredictionModel`$\langle \mathcal{O}, \mathcal{S}_0^n, k, \varepsilon, \sigma_{\min}, \kappa, \mathbf{N} \rangle$

1 **Input:** *(i)* Axioms and entailments $\mathcal{O} : \langle \mathcal{T}, \mathcal{A}, \mathcal{G} \rangle$, *(ii)* The ontology stream $\mathcal{S}_0^n$, *(iii)* The snapshot for prediction k, *(iv)* The lower limit $\varepsilon \in (0, 1]$, *(v)* The minimum drift significance $\sigma_{\min}$, *(vi)* The proportion κ of snapshots with concept drifts used for modelling, *(vii)* The expected number of snapshots with concept dirfts $\mathbf{N}$.
2 **Result:** f: Model for prediction at time k of $\mathcal{S}_0^n$.
3 **begin**
4 $\quad \mathcal{S}_0^n|_\kappa \leftarrow \{k-1\}$; % *Initial snapshot set for learning model.*
5 $\quad$ % *Computation of the most significant drifts w.r.t.* ε, $\sigma_{\min}$.
6 $\quad \mathbb{S} \leftarrow$ `SignificantDrift`$\langle \mathcal{O}, \mathcal{S}_0^{\mathtt{n}}, \varepsilon, \sigma_{\min} \rangle$;
7 $\quad$ % *Selection of* $^{\kappa}/_{\mathbf{N}}$ *snapshots involved in concept drifts* $\mathbb{S}$.
8 $\quad$ **foreach** $i \in [0, k)$ *s.t.* $(i, i{+}1) \in \mathbb{S} \ \wedge \ |\mathcal{S}_0^n|_\kappa| < {}^{\kappa}/_{\mathbf{N}}$ **do**
9 $\qquad \mathcal{S}_0^n|_\kappa \leftarrow \mathcal{S}_0^n|_\kappa \cup \{i\}$;
10 $\quad$ % *Expand* $|\mathcal{S}_0^n|_\kappa|$ *with snapshots not involved in* $\mathbb{S}$.
11 $\quad$ add $^{1-\kappa}/_{\mathbf{N}}$ point(s) of time i to $\mathcal{S}_0^n|_\kappa$ s.t. $(i, i{+}1) \notin \mathbb{S}$;
12 $\quad$ Learning model f using (36) with weight (37) or (38) :

$$\text{(i)} \quad \min_{(a,b) \in \mathbb{R}^{\mathbf{N}} \times \mathbb{R}} \sum_{i=1}^{\mathbf{N}} \omega_{ij} L(\boldsymbol{v}(g_i), f(\boldsymbol{x}_i)) + \alpha R(a)$$

$$\text{(ii)} \quad f(\boldsymbol{x}_i) = a^T \boldsymbol{x}_i + b$$

$\quad$ **return** f;

Algorithm 1 and 3 are parameterized with low ε, $\sigma_{\min}$, high κ and (37) as weight (Line 12) favours models with significant concept drifts for prediction, which supports diversity and prediction changes in the model. Parameterized with high ε, $\sigma_{\min}$, low κ and (38) as weight, it will capture more consistent models.

The linear scoring function f in (36) has the following advantages compared to more complex ML models such as Adaptive-Size Hoeffding Tree [23]: *(i)* better handling over-fitting with reduced sample size — due to filtering out snapshots with no significant concept drifts (Line 8-9 in Algorithm 3), *(ii)* ensuring efficient, scalable learning and prediction for online contexts. According to [48], the optimization algorithm SGD can be adopted to learn the parameters of the linear score function, while the hyperparameter setting can be adjusted by the same approach as in entailment weight learning except that the target is changed from minimizing the loss to minimizing the testing error on a developing sample set.

6. Experimental Results

6.1. Experiment Settings

We evaluated our method by *(i)* studying the impact of semantic reasoning and embeddings on concept drifts for two applications: air quality forecasting in Beijing, China (Beijing for short) and bus delay forecasting in Dublin, Ireland (Dublin for short), and *(ii)* comparing its results with state-of-the-art approaches. The experiments are conducted on: 16 Intel(R) Xeon(R) CPU E5-2680, 2.80GHz cores, 32GB RAM. The source codes[3], the Dublin data[4] and the Beijing data[5] are available for the reproducibility purpose.

• **Beijing Context:** The air quality level in Beijing, ranging from Good (value 5), Moderate (4), Unhealthy (3), Very Unhealthy (2), Hazardous (1) to Emergent (0) is forecasted using data streams of B_1: air pollutants and meteorology elements, B_2: wind speed, and B_3: humidity observed in 12 sites (observations from surrounding cities of Beijing are utilized). The semantics of context is based on a DL $\mathcal{ALC}$ ontology whose TBox includes 48 concepts, 13 roles, 598 axioms. An average of $6,500$ RDF triples (as ABox assertions) are generated at each update (snapshot) for the streams, while one snapshot lasts 600 seconds.

• **Dublin Context:** The bus delay level in Dublin, classified as Free (value 4), Low (3), Moderate (2), Heavy (1), Stopped (0) can be forecasted using reputable live stream contextual data related to D_1: bus GPS location, delay, congestion status, D_2: weather conditions, D_3: road incidents. Table 3 captures the descriptions of the latter data streams, described along the raw data size per day in Mb, their throughput, i.e., frequency of update, the number of ontological axioms received per update, and their size when serialised in RDF triples. The size of an update in RDF is much larger than the number of axioms, as each axiom could require up to 12 triples, after serialisation. This is particularly the case for DL $\mathcal{EL}^{++}$ as blank nodes are required for existential quantification. We consider an extended setting by enriching data using a DL $\mathcal{EL}^{++}$ domain ontology. The ontology (TBox and ABox) ultimately includes 55 concepts, 19 roles and $25,456$ axioms.

---

[3] `https://bit.ly/36KOxOP`, and `https://goo.gl/TXdMpv`

[4] `https://bit.ly/30HeOK6`

[5] `http://bit.ly/2c8KmfZ`

Feature DataSet	Size (Mb) per day	Frequency of Update (seconds)	# Axioms per Update	# RDF Triples per Update
D_1: Bus	120	40	3,000	12,000
D_2: Weather	3	300	53	318
D_3: Incident	0.1	600	81	324

Table 3: Dataset Details of the Dublin Bus Delay Context.

The RDF descriptions of data streams for both Beijing and Dublin contexts are represented through the vocabulary of their respective domain ontologies. An internal lightweight triple-based representation has been adopted to cope with scale and OWL/RDF mapping.

• **Knowledge Graph Embedding Setting**: The results are reported with the following setting to Algorithm 2. SGD is adopted as the optimizer. The hyperparameters b, μ and r are set to 16, 0.005 and 0.02. m is set to ${}^{n}/_{b} \times 5$ where n represents the size of training samples. Such hyperparameters have been experimented to be the best (with respect to the downstream task of stream classification) for both Beijing and Dublin contexts. We do not report an extensive evaluation of hyperparmeters configurations and their respective results on those contexts as they are use case specific. We focus more on experimenting the impact of the knowledge graph embeddings on state-of-the-art approaches for solving the stream classification task. Fine tuning hyperparameters does have an impact, but our evaluation aims at emphasizing on the impact of integrating such sophisticated embeddings (Figure 6, Figure 7), and understanding the best context and fine-tuning for optimal results (Table 4, Figure 3, Figure 4, Figure 5).

6.2. Classification Tasks

Two classification tasks have been studied under different contexts of semantic expressivity, ontology size, and stream throughputs.

• **Classification Task in Beijing Context:** The classification task is to predict air quality in Beijing where data is exposed through three sources of streaming data with semantic representations in DL $\mathcal{ALC}$. The number of classes is five for the classification problem. The context variation, characterizing a concept drift problem, makes the air quality difficult to be forecasted.

• **Classification Task in Dublin Context:** The task is to predict bus delay in Dublin where data is exposed through three sources of streaming data with semantic representations in DL $\mathcal{EL}^{++}$. The number of classes is four for the classification problem. Bus delay is subject to major changes due the high degree of context variation. The latter, responsible for the concept drift problem, impacts accuracy the most.

• **Metric:** We used the macro-averaged F1-score (macro F1-score for short) as a proxy for evaluating the classifiers overall accuracy. All scores reported in Table 4, Figures 3, 4, 5, 6 and 7 are Macro F1-scores as it is widely used for evaluating classifiers operating on more than two classes.

• **Validation:** The macro-F1 score is measured by comparing the predicted entailments with the ground truth entailments which are observed or inferred from the observations. A cutting time is used to split the streaming data: observations before the cutting time are used for training, while those after the cutting time are used for evaluation. Three different cutting times are set for three different splits in each context. Multiple duplicated tests are conducted for each split. We used the average as final result. Note given a training split, we further randomly extract some samples as the validation set for searching suitable hyperparameter settings.

6.3. Evaluation of Knowledge Graph Embeddings

In this section we evaluate the impact of knowledge graph embeddings, described as a core element to capture and compact entailment and consistency representations of temporal snapshots in a stream. The evaluation aims at demonstrating that such embeddings explicit the semantics of stream dynamics, and then provide added-value (i.e., better macro-F1 scores) when applied to solve a classification tasks in a stream context with significant concept drifts. In particular we studied how our approach is robust to significant concept drift. The significance of concept drift is characterized by the proportion of stream updates. Our approach is evaluated with up to 51% of stream updates.

• **Overall Semantic Impact:** Table 4 reports the positive impact of using knowledge graph embeddings (cf. columns with ✓and ✓✓) on all forecasting tasks in both Beijing and Dublin contexts. The first four rows are results on the Beijing context where different data streams have been used: B_1, B_2 and B_3 (cf. Section 6.1 for details on context). The last four rows are results on the Dublin context with data streams D_1, D_2 and D_3 from Table 3. The row on average improvement refers to the gain from using knowledge graph embeddings with two different encodings — BOE and WBOE, when compared to using no such embeddings. When referring to knowledge graph embeddings (cf. columns with ✗) we refer to the embedding procedure as described in Algorithm 2 (which captures entailment vectors, weights and consistency vector). The third, sixth and ninth columns refer to no encoding with ω_{ij} is set to 1 and $\mathcal{S}_0^n|_\kappa$ is set to all snapshots of $\mathcal{S}_0^n$ before time k in Algorithm 3. The fourth, seventh and tenth columns refer to knowledge graph embedding with BOE. The fifth, eighth and eleventh refer to knowledge graph embedding with WBOE. The improvements on the Macro-F1 score due to using knowledge graph embeddings of either BOE or WBOE are significant, ranging from 12.8% (12.5%) to 35.7% $(34.1\%$ resp.) in the Beijing (Dublin resp.) Context. The embeddings naturally identify semantically (dis-)similar contexts by capturing temporal (in-)consistency. Thus, they help in building discriminating models, even for long-term-ahead forecasting, as shown for Δ = 18-hours with a 35.7% $(34.1\%$ resp.) gain in average in the Beijing (Dublin resp.) Context.

• **BOE vs. WBOE Encoding:** Table 4 presents that learning entailment weights (Algorithm 2) improves the knowledge graph embeddings and leads to more significant improvements on the macro-F1 score. WBOE (columns ✓✓) outperforms BOE (columns ✓) by (4.2%, 12.7%, 13.0%) in the Beijing context, and by (11.0%, 15.8%, 17.9%) in the Dublin context when Δ is set to (6, 12, 18). The results demonstrate the positive impact of entailment weights on macro-F1 scores. The output of Algorithm 2 shows over 80% of the entailments in both Beijing and Dublin Contexts are insignificant (with absolute value of weight being less than 0.1). This means only a small part of the entailments play an important role in determining the consistency between two snapshots. In our empirical study, WBOE encoding can be approximated (achieving close macro-F1 scores) by BOE encoding with a manually selected subset of entailments, which however *(i)* needs an exponential number of tests w.r.t. entailment number and *(ii)* is not generic. WBOE encoding extends BOE encoding with an optimized parameterization of the entailment weights.

• **Feature Impact:** Table 4 emphasizes extra macro-F1 score gains when increasing the number of features (data streams) as the input of the approach. For example, the average gain of the macro-F1 score from 1 feature to 3 features is 68.5%. This means incorporating semantic concept drifts and knowledge graph embeddings does not impact the effectiveness of adding input features. Their benefits are independent.

• **Concept Drift Significance:** concept drifts are characterised by 48% and 51% of stream updates in respectively Beijing Context and Dublin Context. We focus on 4 levels of concept drifts, ranging from a $.2$ to $.8$ significance for any Δ in $\{6, 12, 18\}$ cf. semantic concept drift significance (value in $[0, 1]$) in Figure 3. Level-0 does not capture any change. Figure 3 reports the proportion of sever-

City	$\mathcal{ID} : Features$	Δ=6 hours			Δ=12 hours			Δ=18 hours		
		✗	✓	✓✓	✗	✓	✓✓	✗	✓	✓✓
Beijing	$\mathcal{B}_1 : B_1$	.351	.381	.399	.344	.398	.443	.261	.310	.346
	$\mathcal{B}_2 : B_1 + B_2$	.398	.432	.451	.350	.404	.452	.279	.334	.370
	$\mathcal{B}_3 : B_1 + B_3$	.421	.495	.513	.373	.410	.466	.282	.339	.382
	$\mathcal{B}_4 : B_1 + B_2 + B_3$	.501	.591	.614	.389	.428	.487	.286	.348	.407
Avg. Improvement (%)		✓: 13.2 ✓✓: 18.5			✓: 12.8 ✓✓: 27.0			✓: 20.1 ✓✓: 35.7		
Dublin	$\mathcal{D}_1 : D_1$	.455	.491	.516	.387	.399	.444	.321	.339	.389
	$\mathcal{D}_2 : D_1 + D_2$	.534	.580	.691	.499	.521	.556	.361	.401	.501
	$\mathcal{D}_3 : D_1 + D_3$	.601	.666	.705	.513	.569	.647	.371	.441	.550
	$\mathcal{D}_4 : D_1 + D_2 + D_3$	.659	.810	.924	.533	.635	.836	.601	.699	.748
Avg. Improvement (%)		✓: 12.5 ✓✓: 24.6			✓: 9.4 ✓✓: 26.9			✓: 12.9 ✓✓: 34.1		

Table 4: Macro-F1 Scores in Beijing and Dublin Contexts without Knowledge Graph Embeddings (✗), with BOE Knowledge Graph Embeddings (✓) and WBOE Knowledge Graph Embeddings (✓✓) (Evaluation of Algorithm 2 and 3).

ity levels in concept drifts for Beijing Context and Dublin Context e.g., 7% are level-$.4$ for Beijing while 19% are level-$.8$ for Dublin. Although macro-F1 scores clearly declined by increasing the severity level of concept drift e.g., from 96% (level-$.2$) to 21% (level-$.8$) in Dublin Context, knowledge graph embeddings has shown to significantly boost macro-F1 scores. More interestingly the more severe concept drift the higher improvement, i.e., (average) 36% to 56% on level-$.4$ to $.8$. Thus integrating semantics is a way forward to build machine learning models which are robust to changes, potential erroneous sensor data and significant concept drifts.

• **Model Consistency Impact:** Figures 4 and 5 report macro-F1 scores of the forecasting tasks on **H**igh and **L**ow **C**oncept **D**rift versions of the Dublin and Beijing problems, noted by HCD and LCD. 85% and 15% of snapshots are impacted by concept drifts respectively in HCD and LCD.

Algorithm 1 and 3 are evaluated with 3 settings of $(\varepsilon, \sigma_{\min}, \kappa)$: *(i)* consistent model with $(.9, .9, .1)$, *(ii)* mixed model with $(.5, .5, .5)$, *(iii)* inconsistent model with $(.1, .1, .9)$. $\mathbf{N} = 1,500$. Figure 4 (resp. 5) reports that prediction with consistent (resp. inconsistent) samples, outperforms models with inconsistent (resp. consistent) samples, by about 318% (resp. 456%) and 254% (resp. 322%) in respectively Beijing and Dublin for LCD (resp. HCD). Prediction with consistent and inconsistent samples corresponds to using (38) and (37) to calculation the sample weight ω_{ij} respectively. These results confirm the importance of *(i)* inferring concept drifts and concept drift significance (by Algorithm 1), and *(ii)* semantic prediction with consistent vectors (by Algorithm 3).

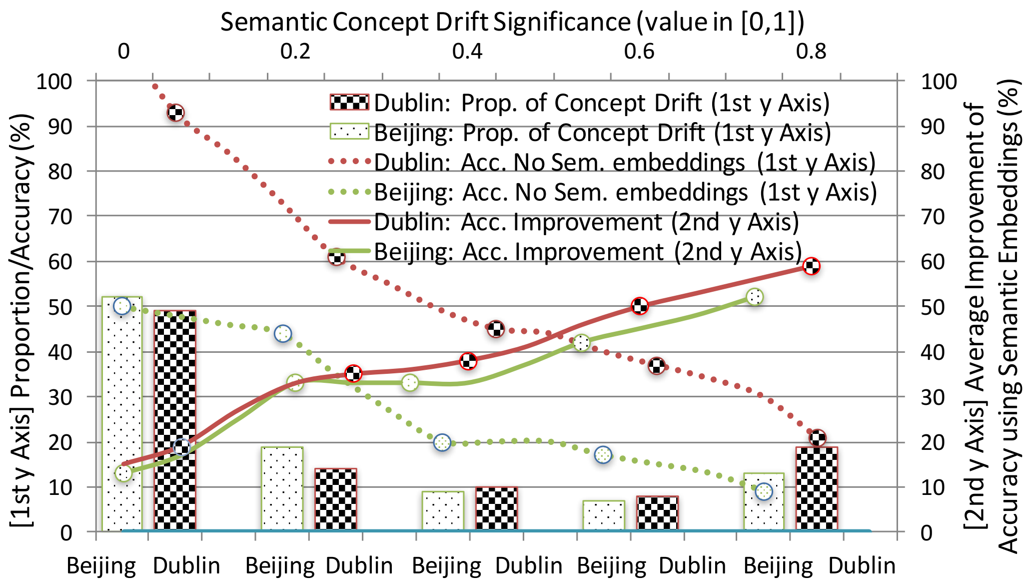

Figure 3: Concept Drift Significance Test: Forecasting Macro-F1 Scores (as a proxy for accuracy) - Evaluation of Algorithm 1 and 3).

6.4. *Comparison with Baselines*

We evaluate our final linear score function trained with the semantic concept drift analysis and the embeddings with the following baselines whose optimum hyperparameters are set via grid searching:

i) **S**tochastic **G**radient **D**escent (SGD) on a neural network perceptron (Hyperparameters for both Beijing and Dublin context: logistic regression loss, L2 regularization, no early stopping, 1 hidden layer, 128 units in the hidden layer, ReLU non-linear function, a learning rate of 0.01). Implementation documentation[6];

ii) **L**ogistic **R**egression (LR) on a linear model (Hyperparameters for both Beijing and Dublin contexts: logistic regression loss, newton-cg solver, L2 regularization). Implementation documentation[7];

[6]https://pytorch.org/docs/stable/nn.html

[7]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

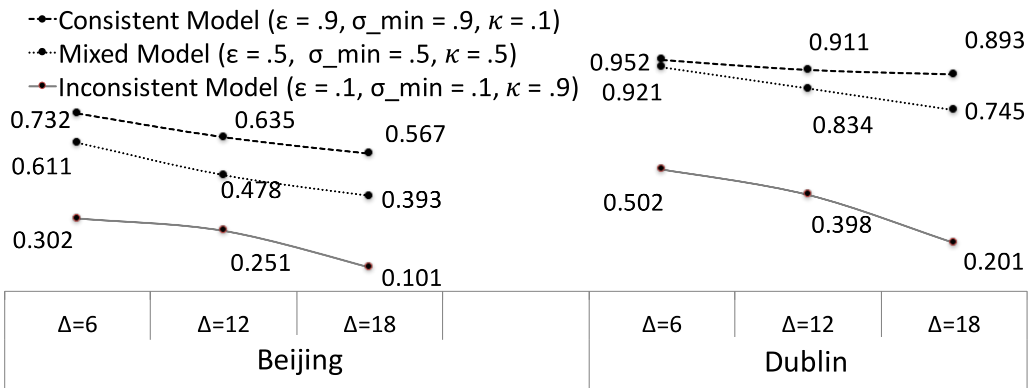

Figure 4: The Impact of Model Consistency on Forecasting the Macro-F1 Scores, for Low Concept Drift Streams where 15% of Snapshots are Impacted by Concept Drifts (Evaluation of Algorithm 1 and 3).

iii) **R**andom **F**orest (RF) (Hyperparameters with number of trees in forest: 104 and 76 for the Beijing context and the Dublin context respectively, max features for splitting a node: 34 and 21 for the Beijing context and the Dublin context respectively, and bootstrap and max number of levels in each decision tree: 18 for both contexts). Implementation documentation[8];

iv) A method addressing concept drifts in stream learning: **A**daptive-**S**ize **H**oeffding **T**ree (ASHT) [23] (Hyperparameters are the default ones for both Beijing and Dublin contexts). Implementation documentation[9];

v) A method addressing concept drifts in stream learning: **L**everaging **B**agging (LB) [26] (Hyperparameters with L2 regularization for both Beijing and Dublin contexts). Implementation documentation[10];

vi) **L**ong **S**hort-**T**erm **M**emory (LSTM), a state-of-the-art Recurrent Neural Network for time-series forecasting [49] (Hyperparameters for the Beijing context: batch size of 32, an architecture of 2 layers, 25 units in each layer, 100 epochs for training, with no dropout; hyperparameters for the Dublin con-

[8]`https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

[9]`https://github.com/scikit-multiflow/scikit-multiflow/`

[10]`https://github.com/scikit-multiflow/scikit-multiflow/`

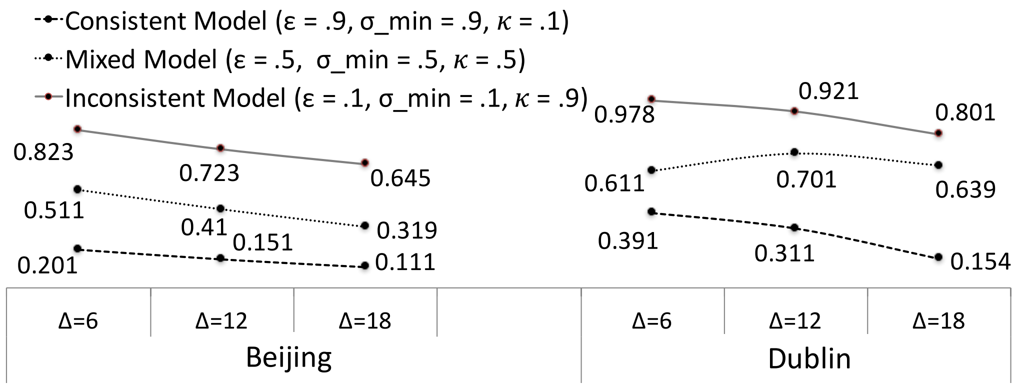

Figure 5: The Impact of Model Consistency on Forecasting the Macro-F1 Scores, for High Concept Drift streams where 85% of Snapshots are Impacted by Concept Drifts (Evaluation of Algorithm 1 and 3).

text: batch size of 32, an architecture of 2 layers, 50 units for each layer, 200 epochs for training, a dropout of 0.25). Implementation documentation[11];

vii) **A**uto-**R**egressive **I**ntegrated **M**oving **A**verage (ARIMA), a standard time-series forecasting model [50] (Hyperparameters for both Beijing and Dublin contexts: disp is set to False, transparams is set to True, trend includes constant, solver is set to newton). Implementation documentation[12];

We adopted the above baselines as they are state-of-the-art for stream learning tasks. They are all approaches that learn a representation in some way, with different methodologies. As we do not want to be biased towards a particular representation learning paradigm we did inclusive experiments with them, and mainly evaluate the impact of semantics, and our knowledge graph embeddings approach on such approaches. To this end, we add semantic component (by Algorithm 1-2) to the machine learning models SGD, LR, RF, ASHT and LB (by Algorithm 3), and compare the semantic enhanced (denoted by prefix S- in Figure 6) with the original as well as LSTM and ARIMA. ARIMA uses one stream variable: The air quality level in Beijing and the bus delay level in Dublin, while SGD, LSTM, ASHT and LB use all the streams with an optimized memory size, i.e., the number of recent snapshots (i.e., $\mathcal{B}_4$ and $\mathcal{D}_4$ in Table 4). Results with optimum parameters

[11] `https://pytorch.org/docs/master/generated/torch.nn.LSTM.html`
[12] `https://www.statsmodels.org/stable/index.html`

for Algorithm 1-3 are reported.

Figures 6 and 7 emphasis that our semantic enhanced models outperformed the baselines, especially in the Dublin Context. S-SGD, S-LR, S-RF, S-ASHT and S-LB outperform their original models by 21.37%, 20.5%, 18.81%, 13.21% and 12.37% (40.0%, 47.3%, 24.6%, 21.6% and 15.0%) respectively w.r.t. the Beijing Context (Dublin Context). Among them, S-SGD which adopts a simple linear score function performs the best. S-ASTH and S-LB, although equipped with both knowledge graph embeddings and statistic learning strategies (cf. Section 2.2) for concept drifts, do not outperform S-SGD. Meanwhile, S-SGD's average macro-F1 score is 7.03% (17.79%) higher than LSTM (ARIMA) in the Beijing Context, and 18.8% (43.6%) higher in the Dublin Context. The enhancement by semantic reasoning and embeddings is more significant in the Dublin Context than in the Beijing Context. One potential reason is the difference of the ontology expressiveness: the Beijing Context adopts DL $\mathcal{ALC}$ while the Dublin Context adopts DL $\mathcal{EL}^{++}$ (cf. Lessons Learnt).

More interestingly, classic learning models do not generalise as well as the models that are enhanced by semantic reasoning and embeddings. The later models show to be more robust with less variance. The experiments also demonstrate that semantic consistency and inconsistency matters more than recentness during learning.

6.5. Lessons Learnt

Adding semantic reasoning and embeddings to classic ML models has clearly shown the positive impact on the macro-F1 scores and the stability, especially in presence of concept drifts. Simple ML models (e.g., the linear score function) which are fast to train can achieve higher macro-F1 scores than complex models (e.g., LSTM) when semantics are added. Our study also finds that the attentions (through weights) to different entailments in comparing two snapshots, w.r.t. a specific target, are different. Learning weighted bag of entailments (WBOE) benefits understanding overall semantics of all the entailments.

Meanwhile, ontology expressiveness and axiom numbers are critical as they drive and control the semantics of data in streams. They determine the entailments that can be inferred, which further impact the knowledge graph embeddings including the entailment vectors, the entailment weights and the consistent vector. The more semantic axioms the streams have, the more robust the model is, and hence the higher macro-F1 score it achieves. Lightweight semantics such as RDF Schema would highly limit the scope of our model given the omission of inconsistency checking (cf. Figures 4-5). On the other hand considering more expressive

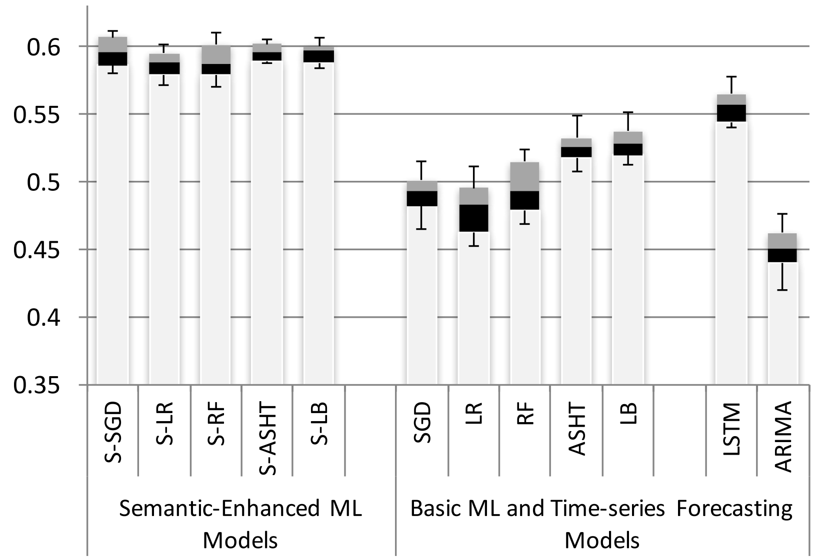

Figure 6: [Beijing Context] Baseline Comparison of Forecasting Macro-F1 Score (Evaluation of Algorithm 1-3), where $\Delta = 6$. Best results are from the semantic enhanced models S-SGD, S-LR, S-RF, S-ASHT and S-LB. In particular S-SGB with a score of $.62$ is achieving the best results overall.

semantics would strongly impact the entailment vector computation, as they require an underlying reasoning to be applied, i.e., entailments rely on reasoning over the TBox.

7. Conclusion and Future Work

In this work, we proposed an approach to encoding knowledge in ontology streams with schema-enabled knowledge graph embeddings, through a novel combinations of ABox entailment vectors, entailment weights and a consistency vector, alongside a general framework of coupling such schema-enabled embeddings with supervised stream learning algorithms to learn prediction models which are robust to concept drifts.

Interestingly, our approach is adaptable and flexible to any machine learning classification algorithms for streaming learning. Our overall prediction algorithm (Algorithm 3) is agnostic to these classification algorithms, which capture the temporal dependencies of snapshots through our proposed schema-enabled knowledge graph embeddings, among which, the construction of entailment vec-

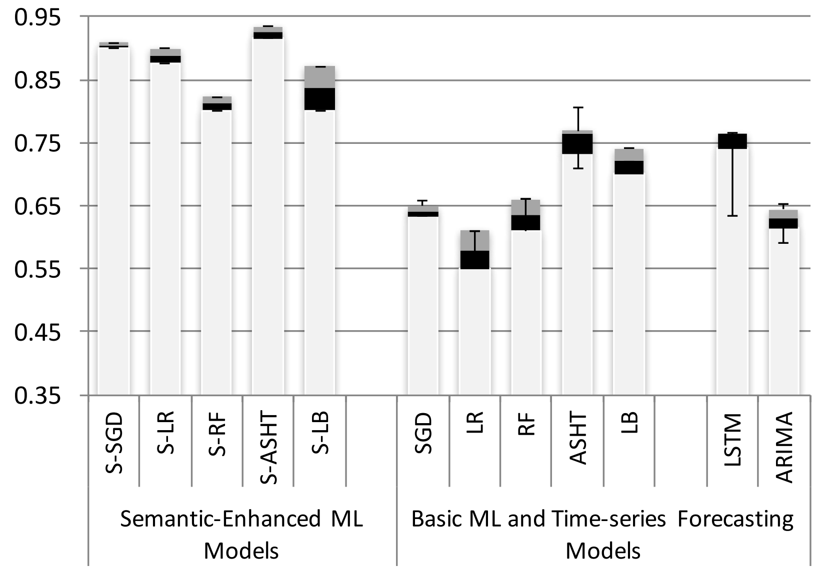

Figure 7: [Dublin Context] Baseline Comparison of Forecasting Macro-F1 Score (Evaluation of Algorithm 1-3), where $\Delta = 6$. Best results are from the semantic enhanced models S-SGD, S-LR, S-RF, S-ASHT and S-LB. In particular S-ASHT with a score of .93 is achieving the best results overall.

tors and consistency vectors is based on ontological reasoning, while the entailment weights are learned iteratively (Algorithm 2).

Another insight is that, in order to check the consistency between two snapshots, only a small part (less than 20%) of ABox entailments play an important role. However, the performance of consistent models and that of inconsistent models have significant difference — the former outperforms the latter by over 300%.

Our work sheds some lights on some of the blind spots of stream learning. Besides demonstrating accurate prediction with concept drifts in Dublin and Beijing forecasting applications, experiments have shown that embedding expressive ontologies with different weights on different entailments is a promising way towards outperforming state-of-the-art approaches.

In the future work, we will *(i)* investigate the impact of ontologies with different levels of expressiveness on on stream learning; *(ii)* extend the approach to incorporate symbolic knowledge (such as ontologies) in other relevant prediction contexts, such as transfer learning and zero-shot learning. Furthermore, our work might be useful for future work on applications of stream learning, such as autonomous driving, which require high accuracy of stream learning in the presence of sudden and disruptive changes .

Acknowledgments

This work is partially funded by the SIRIUS Centre for Scalable Data Access (Research Council of Norway, project 237889), Samsung Research UK, NSFC 91846204, and the EU Marie Curie IAPP K-Drive project (286348).

[1] J. Chen, H. Chen, D. Hu, J. Z. Pan, Y. Zhou, Smog disaster forecasting using social web data and physical sensor data, in: 2015 IEEE International Conference on Big Data (Big Data), IEEE, 2015, pp. 991–998.

[2] D. W. Cheung, J. Han, V. T. Ng, C. Wong, Maintenance of discovered association rules in large databases: An incremental updating technique, in: Data Engineering, 1996. Proceedings of the Twelfth International Conference on, IEEE, 1996, pp. 106–114.

[3] H. M. Gomes, J. P. Barddal, F. Enembreck, A. Bifet, A survey on ensemble learning for data stream classification, ACM Computing Surveys (CSUR) 50 (2) (2017) 23.

[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Computing Surveys (CSUR) 46 (4) (2014) 44.

[5] J. Coble, D. J. Cook, Real-time learning when concepts shift, in: Proceedings of the 13th International Florida Artificial Intelligence Research Society Conference, 2000, pp. 192–196.

[6] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, B. Pfahringer, Efficient online evaluation of big data stream classifiers, in: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 59–68.

[7] Z. Huang, H. Stuckenschmidt, Reasoning with multi-version ontologies: A temporal logic approach, in: International Semantic Web Conference, 2005, pp. 398–412.

[8] Y. Ren, J. Z. Pan, Optimising Ontology Stream Reasoning with Truth Maintenance System, in: Proc. of the ACM Conference on Information and Knowledge Management (CIKM 2011), 2011.

[9] Y. Ren, J. Z. Pan, I. Guclu, M. Kollingbaum, A Combined approach to Incremental Reasoning for EL Ontologies, in: Proc. of the 10th International Conference on Web Reasoning and Rule Systems (RR2016), 2016.

[10] F. Lécué, J. Z. Pan, Predicting knowledge in an ontology stream, in: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, 2013, pp. 2662–2669.

[11] F. Lécué, J. Z. Pan, Consistent knowledge discovery from evolving ontologies, in: Proceedings of the 29th Conference on Artificial Intelligence, AAAI, Vol. 15, 2015.

[12] J. Du, J. Z. Pan, S. Wang, K. Qi, Y. Shen, Y. Deng, Validation of growing knowledge graphs by abductive text evidences, in: Proc. of 33rd AAAI Conference on Artificial Intelligence (AAAI 2019), 2019.

[13] J. Chen, F. Lecue, J. Pan, I. Horrocks, H. Chen, Knowledge-based transfer learning explanation, in: 16th International Conference on Principles of Knowledge Representation and Reasoning, AAAI Press, 2018, pp. 349–358.

[14] F. Lécué, J. Chen, J. Z. Pan, H. Chen, Augmenting transfer learning with semantic reasoning, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, 2019, pp. 1779–1785.

[15] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE transactions on pattern analysis and machine intelligence 35 (8) (2013) 1798–1828.

[16] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: NAACL, 2019, pp. 4171–4186.

[17] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Transactions on Neural Networks 20 (1) (2008) 61–80.

[18] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Transactions on Knowledge and Data Engineering.

[19] W. Zhang, B. Paudel, L. Wang, J. Chen, H. Zhu, W. Zhang, A. Bernstein, H. Chen, Iteratively learning embeddings and rules for knowledge graph reasoning, in: The World Wide Web Conference, 2019, pp. 2366–2377.

[20] L.-J. Cao, F. E. Tay, Support vector machine with adaptive parameters in financial time series forecasting, IEEE Transactions on Neural Networks 14 (6) (2003) 1506–1518.

[21] W. Chu, M. Zinkevich, L. Li, A. Thomas, B. Tseng, Unbiased online active learning in data streams, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 195–203.

[22] A. Bifet, R. Gavalda, Learning from Time-Changing Data with Adaptive Windowing, in: SIAM International Conference on Data Mining, Vol. 7, 2007, pp. 443–448.

[23] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 139–148.

[24] J. Gao, W. Fan, J. Han, S. Y. Philip, A general framework for mining concept-drifting data streams with skewed distributions, in: Proceedings of the 2007 SIAM International Conference on Data Mining, 2007, pp. 3–14.

[25] J. Z. J. Kolter, M. A. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, The Journal of Machine Learning Research 8 (2007) 2755–2790.

[26] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part I, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 135–150.

[27] A. Margara, J. Urbani, F. Van Harmelen, H. Bal, Streaming the web: Reasoning over dynamic data, Web Semantics: Science, Services and Agents on the World Wide Web 25 (2014) 24–44.

[28] H. Beck, M. Dao-Tran, T. Eiter, Equivalent stream reasoning programs, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2016, pp. 929–935.

[29] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, M. Grossniklaus, C-sparql: Sparql for continuous querying, in: Proceedings of the 18th international conference on World wide web, ACM, 2009, pp. 1061–1062.

[30] Y. Ren, J. Z. Pan, Optimising ontology stream reasoning with truth maintenance system, in: Proceedings of the 20th ACM international conference on Information and knowledge management, ACM, 2011, pp. 831–836.

[31] E. Thomas, J. Z. Pan, Y. Ren, TrOWL: Tractable OWL 2 reasoning infrastructure, in: Proceedings of the Extended Semantic Web Conference, 2010.

[32] G. Mehdi, E. Kharlamov, O. Savković, G. Xiao, E. G. Kalaycı, S. Brandt, I. Horrocks, M. Roshchin, T. Runkler, Semantic rule-based equipment diagnostics, in: International Semantic Web Conference, Springer, 2017, pp. 314–333.

[33] S. Klarman, T. Meyer, Prediction and explanation over dl-lite data streams, in: International Conference on Logic for Programming Artificial Intelligence and Reasoning, Springer, 2013, pp. 536–551.

[34] F. Lécué, Scalable maintenance of knowledge discovery in an ontology stream, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, 2015, pp. 1457–1463.

[35] P. Ristoski, H. Paulheim, Rdf2vec: Rdf graph embeddings for data mining, in: International Semantic Web Conference, Springer, 2016, pp. 498–514.

[36] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data, in: Proceedings of the 28th international conference on machine learning (ICML-11), 2011, pp. 809–816.

[37] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in neural information processing systems, 2013, pp. 2787–2795.

[38] J. Hao, M. Chen, W. Yu, Y. Sun, W. Wang, Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1709–1719.

[39] H. Paulheim, H. Stuckenschmidt, Fast approximate a-box consistency checking using machine learning, in: European Semantic Web Conference, Springer, 2016, pp. 135–150.

[40] M. Kulmanov, W. Liu-Wei, Y. Yan, R. Hoehndorf, El embeddings: geometric construction of models for the description logic el++, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, 2019, pp. 6103–6109.

[41] J. Chen, P. Hu, E. Jimenez-Ruiz, O. M. Holter, D. Antonyrajah, I. Horrocks, Owl2vec*: Embedding of owl ontologies, arXiv preprint arXiv:2009.14654.

[42] F. Baader, S. Brandt, C. Lutz, Pushing the el envelope, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence, 2005, pp. 364–369.

[43] Y. Ren, J. Z. Pan, K. Lee, Parallel ABox Reasoning of EL Ontologies, in: Proc. of the First Joint International Conference of Semantic Technology (JIST 2011), 2011.

[44] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2000, pp. 71–80.

[45] J. Gao, W. Fan, J. Han, On appropriate assumptions to mine data streams: Analysis and practice, in: Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on, 2007, pp. 143–152. doi:10.1109/ICDM.2007.96.

[46] L. Bottou, Stochastic gradient descent tricks, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 421–436.

[47] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[48] T. Zhang, Solving large scale linear prediction problems using stochastic gradient descent algorithms, in: Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 116.

[49] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[50] J. L. M. Saboia, Autoregressive integrated moving average (arima) models for birth forecasting, Journal of the American Statistical Association 72 (358) (1977) 264–270.